Partial Duplicate Detection for Large Book Collections

Ismet Zeki Yalniz
Dept. of Computer Science
University of Massachusetts
Amherst, MA, 01003
zeki@cs.umass.edu

Ethem F. Can
Dept. of Computer Science
University of Massachusetts
Amherst, MA, 01003
efcan@cs.umass.edu

R. Manmatha
Dept. of Computer Science
University of Massachusetts
Amherst, MA, 01003
manmatha@cs.umass.edu

ABSTRACT

A framework is presented for discovering partial duplicates in large collections of scanned books with optical character recognition (OCR) errors. Each book in the collection is represented by the sequence of words (in the order they appear in the text) which appear only once in the book. These words are referred to as "unique words" and they constitute a small percentage of all the words in a typical book. Along with the order information the set of unique words provides a compact representation which is highly descriptive of the content and the flow of ideas in the book. By aligning the sequence of unique words from two books using the longest common subsequence (LCS) one can discover whether two books are duplicates. Experiments on several datasets show that DUPNIQ is more accurate than traditional methods for duplicate detection such as shingling and is fast. On a collection of 100K scanned English books DUPNIQ detects partial duplicates in 30 min using 350 cores and has precision 0.996 and recall 0.833 compared to shingling with precision 0.992 and recall 0.720. The technique works on other languages as well and is demonstrated for a French dataset.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.7 [**Digital Libraries**]: Collection, Systems Issues

General Terms

Algorithms, Experimentation

Keywords

partial duplicate detection, sequence matching, unique words

1. INTRODUCTION

This paper describes an approach to finding partial duplicates of documents. The focus here is on long noisy documents. By long it is meant book length rather than short messages, records, or short documents of the type seen on the web. The particular application here is to scanned and recognized books from collections such as the Internet Archive or Google Books but it also applies to government, company and legal documents. Partial duplicates help us to explore the relationships between books - for example which ones are multiple editions or versions of a given book.

Books are different from traditional web documents or the kinds of documents used in TREC. Different editions or versions of books are often not straight copies or near duplicates but may show a lot of variation. Such variations include new prefaces or introductions, additional footers and headers and formatting changes. Two editions of Shakespeare's Othello may differ substantially. For example, one may contain only the main text while a second variorum edition may contain only 30% of the main text on the page with the rest being footnotes. In addition there may be spelling changes and OCR errors. To simplify matters we will refer to this process of finding editions and versions as one of finding partial duplicates. Note that a near duplicate or an exact duplicate is a special case of a partial duplicate. It is not common to find near duplicates in books - both because of OCR errors and the addition of new material.

Books must preserve the linear progression of ideas for them to be valid duplicates. Re-ordering the chapters of a book such as Twain's Huckleberry Finn would either not make it meaningful or make it a different book. This structure is preserved even if material is inserted in the form of footnotes or a preface is inserted. One approach, therefore, to finding a duplicate is to consider each book as a sequence of words and do a full alignment. This has been suggested for various tasks such as finding plagiarized portions of documents [8, 11]. However, methods to fully align two texts are expensive even for papers (let alone books) and do not scale well to large collections. Most plagiarism detection techniques instead use a prefiltering stage which involves chunk overlap to detect possible duplicates before running a full alignment to detect which portions are duplicated [11].

We propose an alternative representations for finding partial duplicates efficiently for books which works for documents in the same language. We exploit the characteristics of language - specifically Zipf's law - to come up with a representation. The crux of the idea is to represent each document using the *sequence of unique words* in the document (not in the collection). By unique words we mean words which occur only once in the document. A sequence is created from the unique words by following their natural

CIKM '11 Glasgow, UK

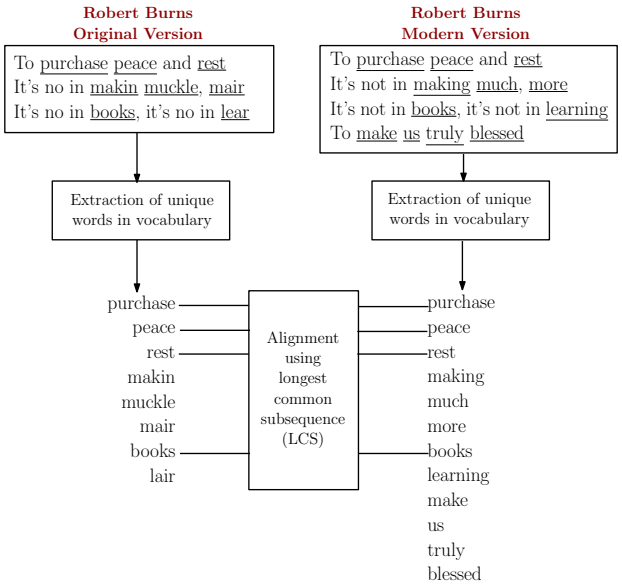


Figure 1: Illustration of partial duplicate detection. Two versions of Robert Burns' poem. Unique words are underlined in the text. The two sequences of unique words are aligned using LCS.

order in the document. Two documents are similar if their sequences are similar [1]. Figure 1 shows two versions of a portion of Robert Burns' poem. For each document, the sequence of unique words is shown from top to bottom.

This representation is compact since unique words are a small percent of all the words in the document. A typical 100K word English book has around 2-3K unique words. Two books are compared by finding the longest common subsequence (LCS) between the two unique word sequences. Since the LCS is done over the sequence of unique words and not over the sequence of all words, the alignment time is drastically reduced. There are additional characteristics of this representation which make the alignment even more efficient and it will be discussed later. For a 100K dataset of books (over 5 billion book pairs), DUPNIQ has a precision of 0.996 and a recall of 0.833. It is more accurate than a 0 mod p shingling technique with p = 50 and 4-grams which has a precision of 0.992 and a recall of 0.720. Our technique is also fast and comparable to shingling in terms of speed. Total processing time for the 100K dataset is 30 min on a 350 core cluster. Experiments over several data sets are also reported including a collection of French books.

There are many uses of such duplicate detection work. The humanities and library communities have a great interest in aggregating works and finding all versions of say Shakespeare's Othello. IFLA's Functional Requirements for Bibliographic Records (FRBR) proposes that the new generation of cataloging systems will include works aggregation [21]. Search engines such as those at the Internet Archive could collapse duplicate books or show the most appropriate work in a set of duplicates - for example an ordinary reader may prefer an uncommented version while a scholar might want a commented version.

[1]Unlike web documents, books don't have the problem of dealing with spammers who might create documents containing only unique words.

In the next section, we characterize the problem of duplicate detection. This is followed by a review of the literature. Section 4 discusses how to find duplicates explaining the unique word representation, the sequence alignment and the scoring used. Datasets and experiments are described next. Finally, we conclude and point to future research directions.

2. PROBLEM CHARACTERIZATION

Our aim here is to find segments of texts which are partial duplicates - for example, this may include an entire book, the main text of a Shakespeare play, a story from a selection of short stories or long excerpts. We exclude quotations or short passages the reason being that a quotation may be used almost anywhere (see [23] on finding quotations). Documents on the same topic (e.g. optics) are not necessarily duplicates so our problem is different from topic detection.

Figure 2 illustrates the variety of duplicates. These images are a selection of duplicates found by the 100K dataset. A full alignment was done on each pair of duplicates. The lengths show the relative lengths of the two book pairs. The books are binned and if the alignment density of a bin is greater than 50% the bin is green, otherwise it is red. Note that the actual play is only a third of the (bottom) Tempest book in Figures 2(b) and 2(c). The figures show that even small portions of duplicates in books can be found even when the metadata gives little hint that the books are duplicates.

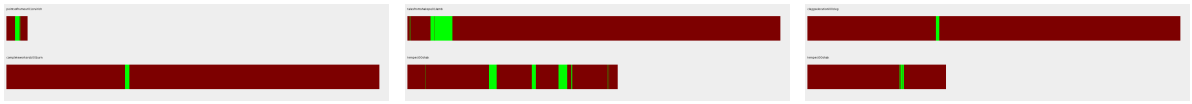
Sometimes one may have two collections of short stories with some overlapping stories. However, the overlapping stories may not have the same order in both books. Note that we just need one short story to do the duplicate detection. Identifying which portion of each book aligns (the full alignment task) is more complicated but that is not the focus of this paper. Certain books like dictionaries, word lists and encyclopedias have an unusually large number of unique words which follow an alphabetical order. This leads to many false matches. Our focus is not on such books.

Note that metadata is not a good solution to finding duplicates since it is often erroneous. Even when correct it is not easy to tell two books are duplicates. Table 1 shows three versions of Shakespeare's Othello. It is not easy to find an automatic algorithm (which works for all books) which will tell from the metadata alone that all contain Shakespeare's Othello. Exact title matches may not also always indicate duplicates. For example, there are two books titled "1914" - one a book of fiction while another is a non-fiction book.

3. RELATED WORK

Several approaches have been previously studied for finding duplicates in different contexts. One well-known application, the Unix program "diff" [17] first creates chunks from each line and then computes the LCS between two sequences of chunks. It works well for comparing different versions of source codes but fails for finding duplicates in scanned book collections because of noise (OCR errors) and variations in line formatting which cause chunks not to be consistent across books.

Most of the work in near duplicate detection involves using either fingerprinting algorithms or using relative frequency techniques (based on using words with similar frequencies) [3]. The fingerprint techniques [3, 5] assume that each document can be broken up into "distinctive" chunks or shingles and two documents which have a large number of chunks



(a) (b) (c)

Figure 2: Examples of duplication. Green bars show matching portions of text with 50% or above word overlap. (a) The book Points of Humour (top) contains a selection of verses from the Complete Works of Robert Burns (bottom). (b) Tales from Shakespeare (top) contains selections from Shakespeare's plays including the Tempest (bottom). (c) Clegg's Elocutionist (top) contains a wide selection of texts including a passage from the Tempest (bottom).

#	Creator	Title
1	No Author	Shakespeare's Tragedy of Othello
2	William Shakespeare and H. H. Furness	A New Variorum Edition of Shakespeare
3	Wiliam Shakespeare, Tommaso Salvini and James Henry Mapleson	Othello: A Tragedy in Five Acts

Table 1: Internet Archive catalogue records for *Othello*

in common are likely to be similar to each other compared to documents which only have a small number in common. Chunks are created by n-grams of words or characters. Note that this is more likely to make the chunks unique since an n-gram is less common than the original word. Hash-keys are generated by hashing the chunks and then indexed using an inverted file (sometimes the chunks are also indexed). Since a document can contain a large number of chunks, most algorithms subsample this set of chunks and differences in sampling strategy distinguish the various approaches [16]. Several sampling techniques have been tried such as full sampling, random sampling, picking every k^{th} chunk [14]. Another approach windows the chunks [22] by picking the chunk with the lowest hash-key as a window is moved over the document. [3] use the fact that every sub-chunk of a duplicated chunk must be non-unique to reduce the number of chunks in multiple passes. Other chunking algorithms include those by [4, 20, 25]. The sub-sampling required in practice means that many of these algorithms do not work as well [3] when the documents are only partial duplicates or noisy OCR output is considered. I-Match [7] extracts words with certain statistical characteristics and hashes their aggregation. These hash values are compared to determine duplication. Talent [9] similarly finds certain kinds of content words and hashes them. Note that both I-Match and Talent use collection statistics rather than individual document statistics as done here. Charikar [6] applied a random projection based method - essentially locality sensitive hashing on the terms of a document - to find near duplicates and Henzinger [15] applied this to the web domain. Hajishirzi et al. [13] also worked on near duplicates by representing each document as a sparse ngram vector and learning the weights depending on the similarity measure being optimized.

Relative frequency techniques assume that two documents with similar words and frequencies must be similar or duplicated [16, 24] and claim to be more accurate than chunking based methods. Local text reuse is a related problem where the duplicates may not be exact. [23] used a Discrete Cosine Transform (DCT) fingerprinting algorithm for this problem.

In plagiarism detection full alignment has been used to find plagiarized passages [8] but is impractical for long documents and large collections. For example, eTBLAST was used to align Medline abstracts [11] but is too computationally expensive for whole documents and instead an essentially chunking based approach was used.

The idea of using sequences of unique words to find duplicates is an extension of Feng and Manmatha's work [12]. Given the OCR output and the ground truth text, Feng and Manmatha find OCR errors with a HMM based alignment approach using unique words as anchors. This approach was later used by Stewart et al. [26] for OCR error correction.

4. FRAMEWORK

The duplicate detection framework DUPNIQ consists of two stages. In the first stage unique words in the document along with their relative positions are extracted from the documents. For efficiency this is done once for every book in the entire collection and thus the process is linear in the number of documents. For a collection consisting of 100K books, it takes about 3.5 min on a cluster of 350 cores to extract and record lists of unique words sorted by their original order in documents. In the second stage the sequences of unique words from two books is compared using an alignment algorithm. If the duplication score is greater than a threshold, the pair is classified as a duplicate. Although there are $d(d-1)/2$ book pairs in a collection of size d, pair-wise comparisons can be done rapidly.

4.1 Extraction of Unique Words

Each book in the collection is represented by the sequence of words (in the order they appear in the text) which appear only once in the context of the book. These words are referred to as "unique words" and they constitute a small percent of all the words in a typical book. The representation is tolerant to OCR errors. While an OCR error may map a non-unique word to a unique word, it is unlikely that two books will have the same corrupted word let alone in the same position in the sequence. Punctuation and numeric characters are removed since the recognition accuracy and consistency is low for them. This also eliminates page numbers which may not be consistent across books but can form a sequence of their own. In addition, the unique words are fairly frequent further reducing the effect of OCR errors and other noise. In a book of 100K words, every second sentence of the document contains a unique word.

4.2 Sequence Matching

Each pair of books in the collection is compared using the sequence of their unique words. The aim is to find the longest subsequence (LCS) of unique words which preserves

Table 2: DUPNIQ-cs and DUPNIQ-its scores for three pairs of books. X and Y refer to the sequences of unique words for Book X and Book Y respectively. GT is the ground truth.

Book X	Book Y	|X|	|Y|	$\|X \cap Y\|$	|LCS|	DUPNIQ-cs	DUPNIQ-its	GT
Shakespeare's Law; S. G. Greenwood	Shakespeare's Law; S. G. Greenwood	1482	1563	1406	1404	0.922	0.979	Yes
Departmental ditties and other verses; R. Kipling	Departmental ditties ballads barrackroom ballads & other verses; R. Kipling	1787	4512	955	739	0.260	0.765	Yes
Metrical Translations; J. Muir	Ausonius	7526	12695	689	53	0.005	0.400	No

the same order in both books. The problem turns out to be a search for the longest common subsequence between the sequences. The length of the LCS is a good indication of duplication between two books (see Figure 1).

There are a number of LCS algorithms in the literature. The standard dynamic programming algorithm has $O(mn)$ time and space requirements, where m and n are the lengths of the two input sequences. There are also algorithms requiring $O(mn)$ time and linear space for computing the LCS. There is also a $O(nloglogn)$ time algorithm with the restriction that no word appears more than once within either input string [18]. This algorithm is actually suitable for the duplication detection case since input sequences are composed of only unique words. For a more detailed discussion on longest common subsequences, please refer to [10].

It is not necessary to compute the LCS length for each book pair in the collection. Given the duplication threshold, it is possible to solve for a lower bound for the LCS length L which makes the book pair a duplicate. If the total number of common unique words between the two sequences is less than L, then the pair is guaranteed to have a duplication score which is below the threshold and therefore the pair can be skipped without any alignment. It is observed that less than 1% of book pairs in the collection pass this test. This test, therefore, provides a considerable speed-up. It is also not necessary to compute the LCS over the entire sequences of unique words. The reason is that a word must appear in both sequences to be in the LCS. Therefore, LCS is run over the sequence of unique words which appear in both sequences [10]. Table 2 shows that the number of common words $X \cap Y$ is much smaller than the unique words in either sequence.

4.3 Scoring Function

The LCS length is used to detect duplicates. However, the LCS length is a function of book length and needs to be normalized. Below two normalization schemes are proposed. Other approaches involving normalizing the LCS length by the minimum or average of the lengths of the two sequences did not work well and therefore, are not reported.

4.3.1 Correlation Score (CS)

The CS score for two (unique word) sequences X and Y is defined by analogy with correlation as:

$$cs(X,Y) = \frac{|LCS(X,Y)|}{\sqrt{(|X| * |Y|)}} \quad (1)$$

where $|LCS(X,Y)|$ is the LCS length of the two sequences. $|X|$ and $|Y|$ denote the length of X and Y respectively. The range of values is [0,1] and the highest score is obtained when the two sequences are identical.

4.3.2 Information Theoretic Score (ITS)

From an information theoretic point of view, the similarity between two objects X and Y maybe defined as [19]:

$$similarity(X,Y) = \frac{\log \Pr(common(X,Y))}{\log \Pr(description(X,Y))} \quad (2)$$

where X and Y are any objects generated by a probabilistic model. According to the formula, two objects are more similar if they share more features. The similarity value is maximized when the two objects are identical.

In our case, X and Y are sequences of unique words. The overlapping content between X and Y is defined by the longest common subsequence:

$$common(X,Y) = LCS(X,Y) \quad (3)$$

and the total information content (description) of X and Y is defined by the alignment of X and Y. Assuming that the probability of any word sequence is inversely proportional to its length, then Eq.2 simplifies as:

$$its(X,Y) = \frac{\log |LCS(X,Y)|}{\log (|X| + |Y| - |LCS(X,Y)|)} \quad (4)$$

The score has a range [0,1] and the maximum value is obtained when the sequences are identical. If $|X|$ and $|Y|$ have no common words, then the score is assumed to be zero.

Table 2 shows two duplicates (first two rows) and a non-duplicate. Duplicates have higher LCS scores and higher DUPNIQ-cs and DUPNIQ-its scores. The thresholds are 0.12 and 0.72 for DUPNIQ-cs and DUPNIQ-its respectively.

5. SYNTHETIC EXPERIMENTS

Synthetic documents are generated for investigating the effect of OCR errors and the amount of overlapping text between two books. A pair of texts is created as follows: A clean (without OCR errors) book from the Project Gutenberg website [2] is used as a reference text. The second text is created by replacing a random portion of the reference text with a sample from another book from the Gutenberg website while ensuring that its length remains the same. A specified amount of character level document noise is added to the second text to simulate OCR errors creating a synthetic document [12]. DUPNIQ-cs and DUPNIQ-its scores are computed between the synthetic and reference book. Experiments are performed 20 times and the scores are averaged. Two different scenarios are experimented: In the first scenario two different books on different topics by different authors are used to generate the reference and the synthetic documents. In the second scenario, the same process is applied using two different novels from the same writer (Joseph A. Altsheler) on the same subject (Civil War). Figure 3

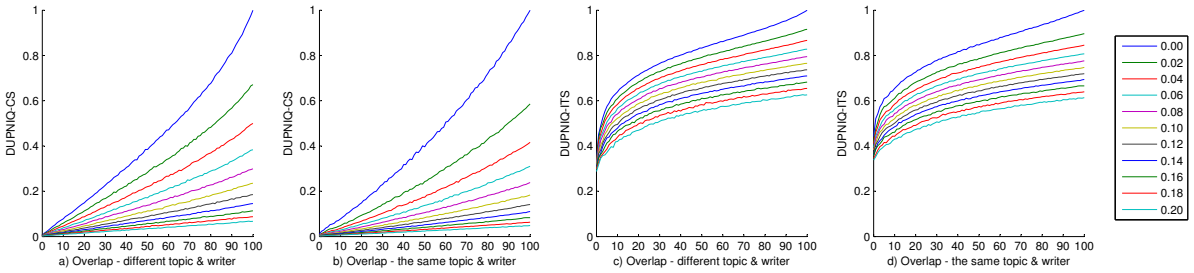


Figure 3: DUPNIQ-its and DUPNIQ-cs versus percentage of overlap and character level document noise. Each line in the plot represents results for the given amount of noise. Note that word error rate is around 75% for 20% character level noise.

shows the duplication scores as the amount of noise and the percentage of text overlap are varied for both scenarios. It is seen that the duplication score does not significantly depend on which scenario is used. DUPNIQ-cs is more sensitive to OCR errors as the amount of noise increases.

6. DUPLICATION EXPERIMENTS

Books from the Internet Archive (IA) [1] were used to construct datasets of various sizes in English and French. The datasets are diverse and include religious books, literary works, technical reports, engineering books, and novels. Language identification was run on all datasets to ensure that the dominant language of each book is as desired. Some of these datasets will be made publicly available at http://ciir.cs.umass.edu/downloads/

The **Training Set** consists of 151 English books containing 67 duplicate pairs labeled manually. This set is used to learn duplicate detection thresholds. The **1K Set** consists of 1,092 English books containing 258 duplicate pairs. The **3K Set** consists of 2,884 French books containing 483 duplicate pairs. The ground truth for 1K and 3K sets are created using a manual technique. The **100K Set** is a large dataset consisting 103,455 English books containing 45,485 duplicate pairs. A major challenge is to estimate the ground truth for this dataset. Metadata alone is not sufficient to create the ground truth since it may be missing, incorrect or incomplete. We made a best effort approach to finding the groundtruth duplicates in this dataset by pooling the outputs of the three different techniques (i.e. DUPNIQ-cs, DUPNIQ-its, and Shingling). Pooling is the strategy used by TREC to judge relevant documents. For pooling we use a lower threshold than the one we estimated on the training set to identify candidate duplicate pairs. Then we do a full alignment on those candidate pairs which is guaranteed to align all duplicate parts. By analyzing the results of the full alignment experiments, we created a subset from the candidate pairs and used the subset as ground truth. In the future we expect to improve the groundtruth on this set. The **Partial Set** has 458 books and tests how well the techniques work on partial duplicates. There are 460 partial duplicate pairs (e.g. ranging from 15% to 80% of a book).

6.1 Evaluation

The threshold is taken to be the value which maximizes the F-measure (the harmonic mean of recall and precision) on the training set. This threshold is used for all datasets including the French dataset.

We considered four baselines - the overlap of unique words (UWO), the overlap of vocabulary words (VO), a 0 mod p Shingling algorithm [3, 20] and I-match. Jaccard similarity is used for the first three. For shingling we tried ngrams with n=2, 4 and 8 and p = 10, 25 and 50. The best shingling results over all datasets were obtained for n=4, p=50 and are reported below. I-match performed poorly. This is understandable given that books are very rarely near duplicates and contain a lot of OCR and other errors. We note that VO - the overlap of vocabulary words - does much better than I-match (see below) indicating that the problem is the global hash function in I-match is sensitive to even small numbers of vocabulary words being wrong (in other words noisy words are found at a variety of IDF values and can't easily be filtered out). We do not discuss I-match further. For the other baselines thresholds are estimated by maximizing the F-measure over the training set as for DUPNIQ.

We provide precision and recall scores for different methods on the datasets in Table 3. For the smaller datasets, DUPNIQ-its did best in terms of F-measure while DUPNIQ-cs did slightly better for the 100K dataset. While shingling did reasonably well on the smaller datasets it did not perform well on the 100K dataset. It is interesting to note that a threshold trained on English books does fine on a French dataset. On the 100K set DUPNIQ-cs and DUPNIQ-its did much better than any other technique in terms of F-measure (DUPNIQ-cs - 0.918, DUPNIQ-its - 0.907, UWO - 0.724, VO - 0.552, Shingling - 0.834). We can understand the performance of the top 3 techniques better by looking at their performance on the Partial dataset. As the results show, both DUPNIQ techniques work better than shingling. DUPNIQ-its, in particular, is a lot better than shingling on partial duplicates. DUPNIQ-its has better performance over DUPNIQ-cs and it is consistent with the synthetic experiments. Between UWO and VO, UWO is the best on most counts. Besides, UWO is much faster since it has many fewer words to deal with.

Results on the 100K set show that some duplicates are missed by DUPNIQ-its and DUPNIQ-cs because their score is slightly lower than the threshold even though they have a high LCS length. This indicates that there is scope for further improvement in length normalization. It may also help to use a larger training set to compute the threshold. 14% of the false negatives are due to dictionaries and encyclopedias. As mentioned before the technique is not meant to work on such books because of their alphabetical ordering. 16% of the false matches are religious book pairs (e.g. gospels, hymns, and prayer books), 12% are literary books, and 6% of them are technical book pairs. The flavor of re-

Table 3: Precision (P) and Recall (R) scores for Training, Small-Test, 1K, 3K, 100K and Partial sets.

Dataset	Training Set		1K Set		3K Set		100k Set		Partial Set	
	P	R	P	R	P	R	P	R	P	R
DUPNIQ-cs	0.971	0.985	**1.000**	0.945	0.940	0.940	0.903	**0.933**	0.989	0.808
DUPNIQ-its	**1.000**	0.985	**1.000**	**0.957**	**0.991**	**0.944**	**0.996**	0.833	**0.995**	**0.919**
UWO	0.956	0.970	0.979	0.937	0.833	0.915	0.593	0.929	-	-
VO	0.970	0.940	0.987	0.887	0.958	0.722	0.440	0.741	-	-
Shingling	0.971	**1.000**	0.983	0.941	0.932	0.892	0.992	0.720	0.963	0.799

sults obtained is similar to that shown in Figure 2. There are also some unusual duplicates. For example, two technical reports published by Johns Hopkins University matched because they contained very similar mailing distribution lists at the end.

7. CONCLUSIONS

The sequence of unique words in a document is proposed as a representation for long documents. In this context, a long document is considered to be one which has at least a few pages of text. The effectiveness of our approach is shown for finding partial duplicates on different datasets in different languages at different scales. Experiments also show that the proposed approach is quite scalable. Although it checks every pair of documents in the collection, partial duplicates can be found in a 100K dataset of books in 30 min on a 350 core cluster with high precision and recall figures. Future work includes further speed-ups, extensions to multiple languages and mapping duplicated portions to each other.

8. ACKNOWLEDGMENTS

We thank James Allan, Bruce Croft and David Smith for discussions and Marek Blat and Logan Giorda for helping with annotations. This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0910884. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

9. REFERENCES

[1] Internet Archive. `http://www.archive.org`, 2010.
[2] Project Gutenberg. `http://www.gutenberg.org`, 2010.
[3] Y. Bernstein and J. Zobel. A scalable system for identifying co-derivative documents. In *SPIRE*, pages 55–67, 2004.
[4] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In *ACM SIGMOD*, pages 398–409, 1995.
[5] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.
[6] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *34th Ann. ACM Symp. on Theory of computing*, pages 380–388, 2002.
[7] A. Chowdhury, O. Frieder, D. A. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, 2002.
[8] P. Clough. Old and new challenges in automatic plagiarism detection. National UK Plagiarism Advisory Service, http://www.ir.shef.ac.uk/cloughie/papers/pas_plagiarism.pdf, 2003.
[9] J. Cooper, A. Coden, and E. Brown. Detecting similar documents using salient terms. In *CIKM*, pages 245–251, 2002.
[10] S. Deorowicz. Solving longest common subsequence and related problems on graphical processing units. *Softw. Pract. Exper.*, 40:673–700, July 2010.
[11] M. Errami, Z. Sun, A. C. George, T. C. Long, M. A. Skinner, J. D. Wren, and H. R. Garner. Identifying duplicate content using statistically improbable phrases. *Bioinformatics*, 26(11):1453–1457, 2010.
[12] S. Feng and R. Manmatha. A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books. In *JCDL*, pages 109–118, 2006.
[13] H. Hajishirzi, W. tau Yih, and A. Kolcz. Adaptive near-duplicate detection via similarity learning. In *SIGIR'10*, pages 419–426, 2010.
[14] N. Heintze. Scalable document fingerprinting. In *USENIX Workshop on Electronic Commerce*, 1996.
[15] M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *ACM SIGIR*, pages 284–291, 2006.
[16] T. C. Hoad and J. Zobel. Methods for identifying versioned and plagiarized documents. *JASIST*, 54(3):203–215, 2003.
[17] J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Technical Report CSTR 41, Bell Laboratories, Murray Hill, NJ, 1976.
[18] J. W. Hunt and T. G. Szymanski. A fast algorithm for computing longest common subsequences. *Commun. ACM*, 20:350–353, May 1977.
[19] D. Lin. An information-theoretic definition of similarity. In *ICML '98*, pages 296–304, 1998.
[20] U. Manber. Finding similar files in a large file system. In *USENIX Winter 1994 Tech. Conf*, pages 1–10, 1994.
[21] D. Mimno, G. Crane, and A. Jones. Hierarchical catalog records: Implementing a FRBR catalog. In *D-Lib Magazine, http://www.dlib.org/dlib/october05/crane/10crane.html*, volume 11, Oct 2005.
[22] S. Schleimer, D. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In *ACM SIGMOD conference*, pages 76–85, 2003.
[23] J. Seo and W. B. Croft. Local text reuse detection. In *ACM SIGIR*, pages 571–578, 2008.
[24] N. Shivakumar and H. Garcia-Molina. Scam: A copy detection mechanism for digital documents. In *Ann. Conf. on the Theory and Practice of Digital Libraries*, 1995.
[25] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents on the web. In *Intl. Workshop on the World Wide Web and Databases*, 1999.
[26] G. Stewart, G. Crane, and A. Babeu. A new generation of textual corpora: mining corpora from very large collections. In *JCDL*, pages 356–365, 2007.